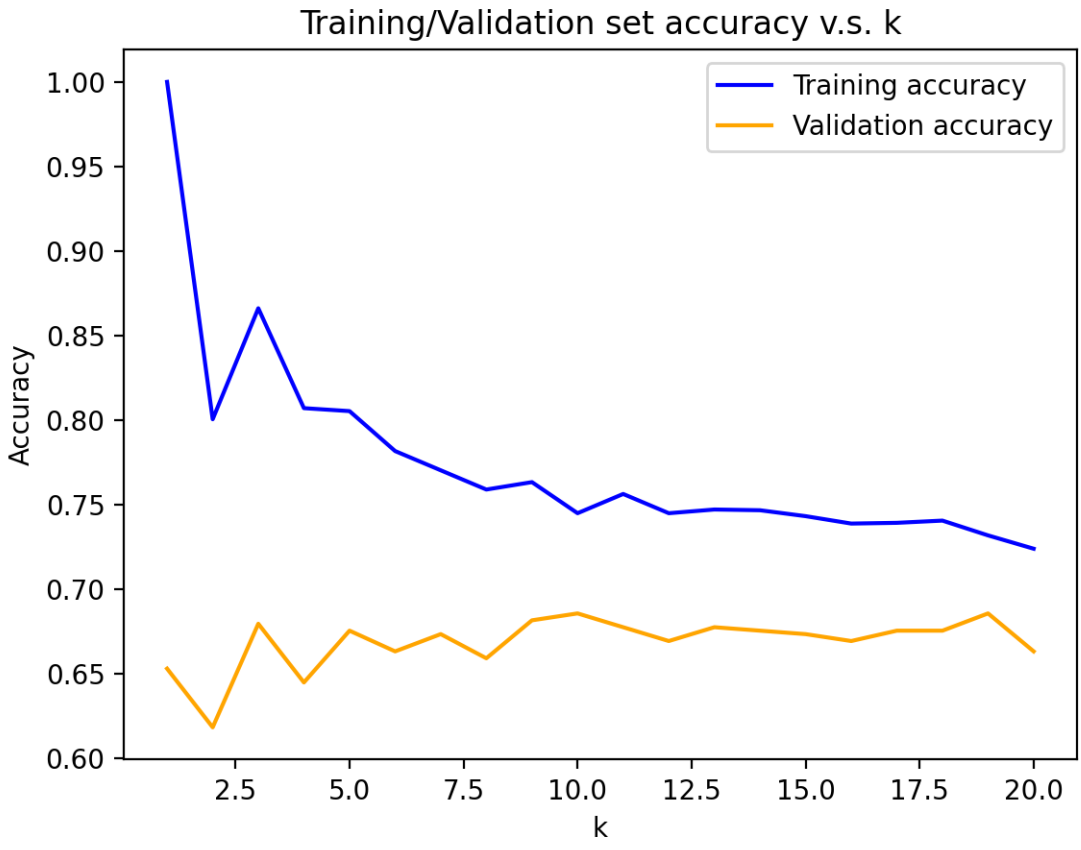
CSC311 Assignment1.pdf

Xinyan He

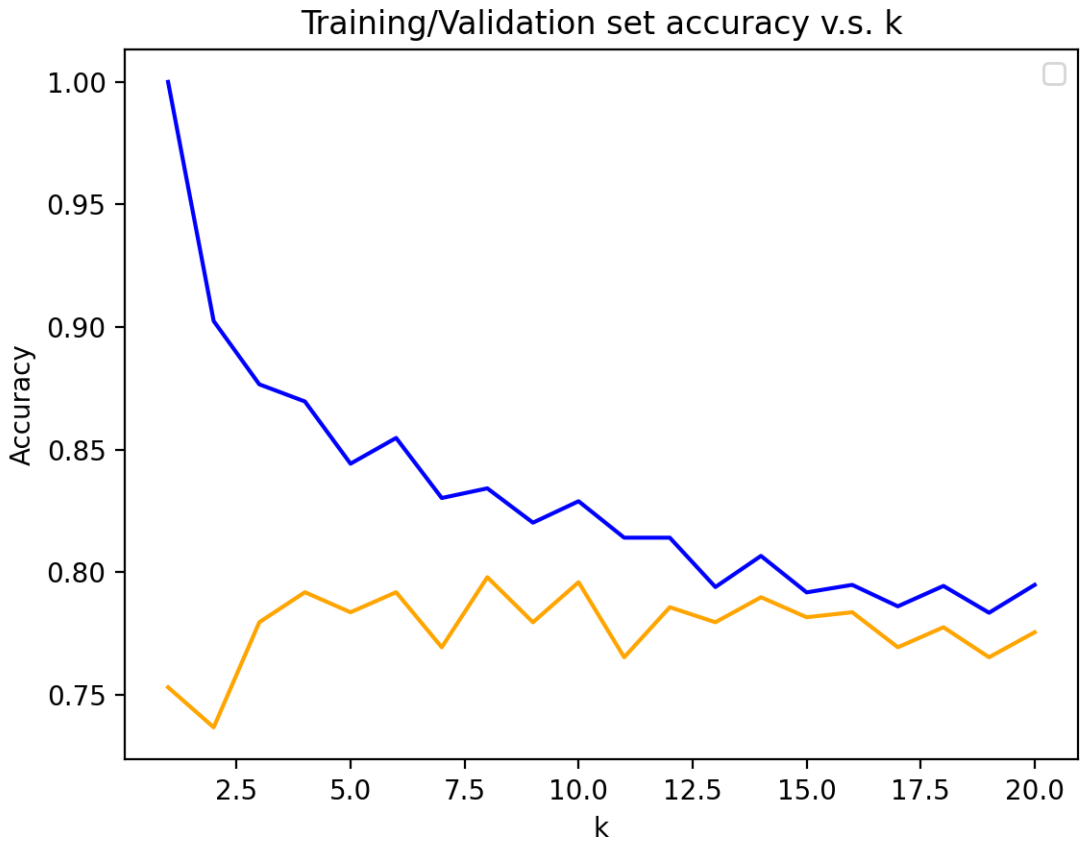
September 30, 2020

1 Q1

(a) The codes are in hw1_q1_code.py
(b) Here I choose the random_state to be 10. At this case: The model with best validation accuracy: KNeighborsClassifier(n_neighbors = 10). The accuracy on the test data is: 0.6265306122448979.



(c) For metric = 'cosine': The model with best validation accuracy: KNeighborsClassifier(metric='cosine', n_neighbors=8). The accuracy on the test data is: 0.7877551020408163. When we are using metrix='cosine', we are measuring the cosine of the angles between these two vector, while Euclidean metric directly measure the distance. It focuses more on the orientation rather than the magnitude compare to Euclidean metric. And here when we are dealing the similarity of words, the magnitude of the vectors does not matter.'cat cat cat' is definitely more similar to 'cat' rather than 'bulldozer' though 'cat cat cat' and 'bulldozer' had a more similar word length. So here we would cosine metric perform better since it focus more on word similarity rather than the magnitude.



2 Q2

(a)First we are trying to calculate $\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial w_j}$

$$\begin{aligned}\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial w_j} &= \frac{\partial \mathcal{J}}{\partial w_j} + \frac{\partial \mathcal{R}}{\partial w_j} \\ &= \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})x_j^{(i)} + \beta_j w_j\end{aligned}$$

$$\begin{aligned}\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial b} &= \frac{\partial \mathcal{J}}{\partial b} + \frac{\partial \mathcal{R}}{\partial b} \\ &= \frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})\end{aligned}$$

$$w_j \leftarrow w_j - \alpha\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial w_j}$$

$$w_j \leftarrow w_j - \alpha(\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})x_j^{(i)} + \beta_j w_j)$$

$$b \leftarrow b - \alpha\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial b}$$

So the final result is:

$$w_j \leftarrow (1 - \alpha\beta_j)w_j - \alpha\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})x_j^{(i)} \tag{1}$$

$$b \leftarrow b - \alpha(\frac{1}{N}\sum_{i=1}^{N}(y^{(i)} - t^{(i)})) \tag{2}$$

We could see after each update of weight, the weight is multiplied by a factor slightly less than 1 which is $1-\alpha\beta_j$ here. This prevents the weights from getting too large which is why it is sometimes called "weight decay".

(b)We will use the following equation to derive the formula:

$$\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial \mathbf{w}_j} = \sum_{j'=1}^{D}\mathbf{A}_{jj'}\mathbf{w}_{j'} - \mathbf{c}_j = 0$$

Based on what we know:

$$\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial \mathbf{w}_j} = \frac{1}{N}\sum_{i=1}^{N}x_j^{(i)}(\sum_{j'=1}^{D}w_{j'}x_{j'}^{(i)} - t^{(i)}) + \beta_j w_j$$

$$\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial \mathbf{w}_j} = \sum_{j'=1}^{D} \frac{1}{N}(\sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)}) w_{j'} - \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} t^{(i)} + \beta_j w_j$$

Finally, we could get

$$\frac{\partial \mathcal{J}_{reg}^{\beta}}{\partial \mathbf{w}_j} = \sum_{j'=1}^{D} [\frac{1}{N}(\sum_{i=1}^{N} x_j^{(i)} x_{j'}^{(i)}) + \delta_{j,j'}] w_{j'} - \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} t^{(i)} = 0 \tag{3}$$

In other words of system,

$$\sum_{j'=1}^{D} A_{jj'} w_{j'} - c_j = 0 \forall j \in \{1, 2...D\}$$

Therefore our final result would be:

$$\mathbf{A}_{jj'} = \frac{1}{N}(\sum_{i=1}^{N} x_j x_{j'}) + \delta_{j,j'} \tag{4}$$

where

$$\delta_{j,j'} = \begin{cases} \beta_j & if j = j' \\ 0 & if j \neq j' \end{cases} \tag{5}$$

and

$$\mathbf{c}_j = \frac{1}{N}\sum_{i=1}^{N} x_j^{(i)} t^{(i)} \tag{6}$$

(c)Determine the formula for $\mathbf{A}$ and $\mathbf{c}$ and derive a closed-form solution for the parameter $\mathbf{w}$. Based on answer in (b),we could get

$$\mathbf{A} = \frac{1}{N}\mathbf{X}^T\mathbf{X} + diag(\boldsymbol{\beta}) \tag{7}$$

Here $\boldsymbol{\beta}$ is a D-dimensional vector.

$$\mathbf{c} = \frac{1}{N}\mathbf{X}^T\mathbf{t} \tag{8}$$

The solution of $\mathbf{w}$ is given by $\mathbf{w} = \mathbf{A}^{-1}\mathbf{c}$ (Assuming $\mathbf{A}$ is invertible), so this gives us the following solution.

$$\mathbf{w} = (\frac{1}{N}\mathbf{X}^T\mathbf{X} + diag(\boldsymbol{\beta}))^{-1}(\frac{1}{N}\mathbf{X}^T\mathbf{t}) \tag{9}$$

The final answer of $\mathbf{w}$ is

$$\mathbf{w} = (\mathbf{X}^T\mathbf{X} + N diag(\boldsymbol{\beta}))^{-1}\mathbf{X}^T\mathbf{t} \tag{10}$$

3 Q3

Here $\mathbf{X}$ is an N $\times$ D matrix. Each row of $\mathbf{X}$ representing a training example. $\mathbf{w}$ is a D-dimensional vector and $\mathbf{1}$ is a N-dimensional vector to denote a vector of all ones.

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b\mathbf{1} \tag{11}$$

Since

$$\frac{\partial \mathcal{J}}{\partial y_i} = \frac{1}{N} sin(y^{(i)} - t^{(i)})$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{y}} = \begin{pmatrix} \frac{\partial \mathcal{J}}{\partial y_1} \\ \vdots \\ \frac{\partial \mathcal{J}}{\partial y_N} \end{pmatrix} = \frac{1}{N} \begin{pmatrix} sin(y^{(1)} - t^{(1)}) \\ \vdots \\ sin(y^{(N)} - t^{(N)}) \end{pmatrix} \tag{12}$$

which can be simplified as

$$\frac{\partial \mathcal{J}}{\partial \mathbf{y}} = \frac{1}{N} sin(\mathbf{y} - \mathbf{t}) \tag{13}$$

As

$$\frac{\partial y_i}{\partial w_j} = x_j^{(i)}$$

We could get

$$\frac{\partial \mathbf{y}}{\partial \mathbf{w}} = \begin{pmatrix} \frac{\partial y_1}{\partial w_1} & \dots & \frac{\partial y_1}{\partial w_D} \\ \vdots & & \\ \frac{\partial y_N}{\partial w_1} & \dots & \frac{\partial y_N}{\partial w_D} \end{pmatrix} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} \dots & x_D^{(1)} \\ \vdots & & \\ x_1^{(N)} & x_2^{(N)} \dots & x_D^{(N)} \end{pmatrix}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = \frac{\partial \mathcal{J}}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{w}} = \frac{1}{N} \cdot \begin{pmatrix} sin(y^{(1)} - t^{(1)}) & \dots & sin(y^{(N)} - t^{(N)}) \end{pmatrix} \cdot \begin{pmatrix} x_1^{(1)} & x_2^{(1)} \dots & x_D^{(1)} \\ \vdots & & \\ x_1^{(N)} & x_2^{(N)} \dots & x_D^{(N)} \end{pmatrix}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = \frac{1}{N} \begin{pmatrix} \sum_{i=1}^{N} sin(y^{(i)} - t^{(i)}) \cdot x_1^{(i)} \\ \vdots \\ \sum_{i=1}^{N} sin(y^{(i)} - t^{(i)}) \cdot x_D^{(i)} \end{pmatrix} \tag{14}$$

$$\frac{\partial \mathcal{J}}{\partial b} = \frac{\partial \mathcal{J}}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial b} = \frac{1}{N} \cdot \begin{pmatrix} sin(y^{(1)} - t^{(1)}) & \dots & sin(y^{(N)} - t^{(N)}) \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$$

$$\frac{\partial \mathcal{J}}{\partial b} = \frac{1}{N} \sum_{i=1}^{N} sin(y^{(i)} - t^{(i)}) \tag{15}$$

To simplify our answer in a sequenced of vectorized mathematical expression with respect to **w** and b, our final answer would be:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + b\mathbf{1} \tag{16}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{y}} = \frac{1}{N} sin(\mathbf{y} - \mathbf{t}) = \frac{1}{N} sin(\mathbf{X}\mathbf{w} + b\mathbf{1} - \mathbf{t}) \tag{17}$$

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}} = \frac{1}{N} X^T sin(\mathbf{y} - \mathbf{t}) = \frac{1}{N} X^T sin(\mathbf{X}\mathbf{w} + b\mathbf{1} - \mathbf{t}) \tag{18}$$

$$\frac{\partial \mathcal{J}}{\partial b} = \frac{1}{N} \mathbf{1}^T sin(\mathbf{y} - \mathbf{t}) = \frac{1}{N} \mathbf{1}^T sin(\mathbf{X}\mathbf{w} + b\mathbf{1} - \mathbf{t}) \tag{19}$$

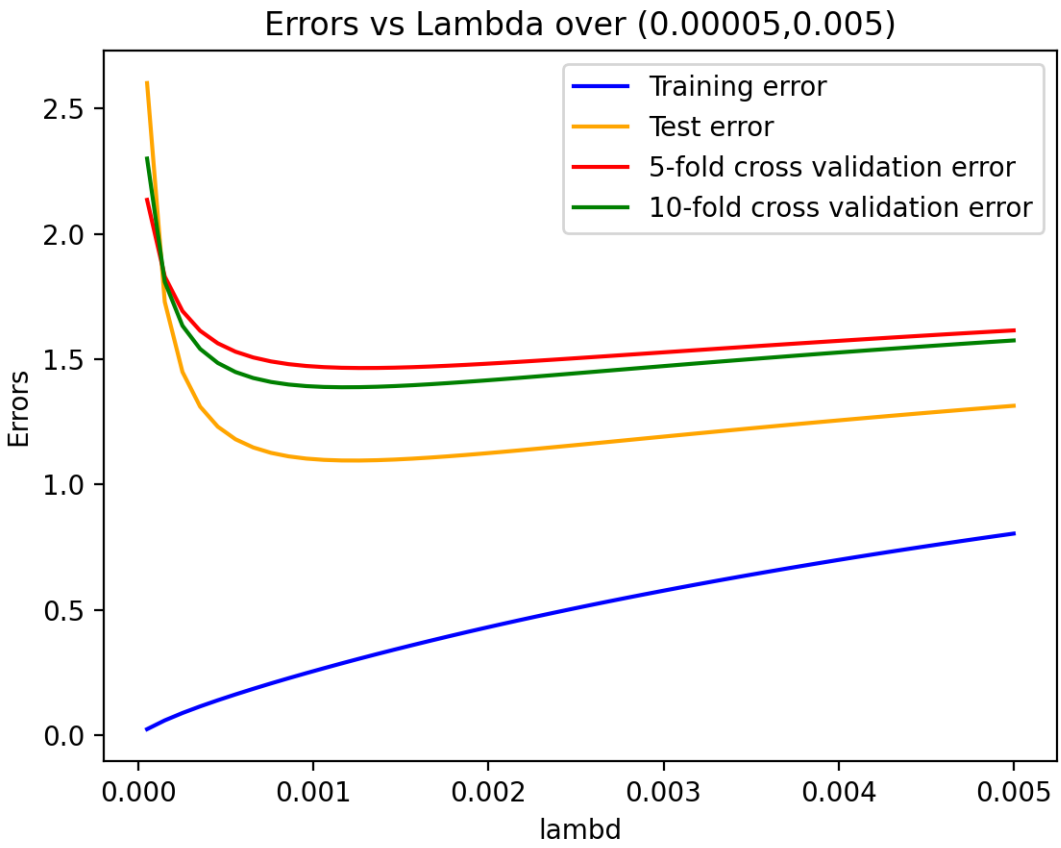
4 Q4

(a) The codes are in hw1_q4_code.py
(b) The correct order and arguments to do cross validation are in hw1_q4_code.py
(c) We set the random seed to 1 here. The training and test errors for all λ in lambd_sequence are as following. The codes are in hw1_q4_code.py

```
[evaluate hw1_q4_code.py]
For lambd 5e-05:
The training errors is 0.023827398142615905. The test error is 2.601391644823132.
For lambd 0.0001510204081632653:
The training errors is 0.05891274573913382. The test error is 1.7287677994041541.
For lambd 0.0002520408163265306:
The training errors is 0.08829510341160733. The test error is 1.449041107721427.
For lambd 0.000353061224489796:
The training errors is 0.11469870650014875. The test error is 1.3106086900637677.
For lambd 0.0004540816326530613:
The training errors is 0.13923988850774655. The test error is 1.230498596164057.
For lambd 0.0005551020408163266:
The training errors is 0.16248341903497196. The test error is 1.180389255472876.
For lambd 0.000656122448979592:
The training errors is 0.18474660580356286. The test error is 1.1477791855291433.
For lambd 0.0007571428571428573:
The training errors is 0.20622199234382155. The test error is 1.126249517067796.
For lambd 0.0008581632653061226:
The training errors is 0.22703389988077496. The test error is 1.1121518828937424.
For lambd 0.0009591836734693879:
The training errors is 0.24726692464648709. The test error is 1.1032544106716156.
For lambd 0.001060204081632653:
The training errors is 0.2669813250794569. The test error is 1.0981135197902825.
For lambd 0.0011612244897959184:
The training errors is 0.2862218047050552. The test error is 1.0957538481482652.
For lambd 0.0012622448979591838:
The training errors is 0.30502276073061607. The test error is 1.0954928990341024.
For lambd 0.001363265306122449:
The training errors is 0.32341154386293675. The test error is 1.0968392292395663.
For lambd 0.0014642857142857144:
The training errors is 0.3414105505221353. The test error is 1.0994304538825799.
For lambd 0.0015653061224489796:
The training errors is 0.35903860424252554. The test error is 1.1029939651442389.
For lambd 0.0016663265306122445:
The training errors is 0.37631189058655873. The test error is 1.10732118947552.
For lambd 0.0017673469387755104:
The training errors is 0.39324460386127696. The test error is 1.1122502216496666.
For lambd 0.0018683673469387756:
The training errors is 0.40984940331483316. The test error is 1.117653811783481.
For lambd 0.0019693877551020413:
The training errors is 0.4261377407022196. The test error is 1.1234308705075882.
For lambd 0.0020704081632653064:
The training errors is 0.44212009936214175. The test error is 1.1295003442370213.
For lambd 0.0021714285714285715:
The training errors is 0.4578061713896183. The test error is 1.135796722480583.
For lambd 0.0022724489795918367:
The training errors is 0.47320499084143136. The test error is 1.1422666911374018.
For lambd 0.0023734693877551023:
The training errors is 0.48832503528124205. The test error is 1.1488666047195633.
For lambd 0.0024744897959183675:
The training errors is 0.5031743042366998. The test error is 1.1555605531143072.
For lambd 0.0025755102040816326:
The training errors is 0.5177603806219528. The test error is 1.1623188662165584.
For lambd 0.0026765306122448983:
The training errors is 0.5320964794538492. The test error is 1.169116945285102.
For lambd 0.0027775510204081635:
The training errors is 0.5461714869921586. The test error is 1.1759343410147305.
For lambd 0.0028785714285714287:
The training errors is 0.5600099925916273. The test error is 1.1827540199582736.
For lambd 0.0029795918367346943:
The training errors is 0.5736123149542308. The test error is 1.1895617761946025.
```

```
For lambd 0.0029795918367346943:
The training errors is 0.5736123149542308. The test error is 1.1895617761946025.
For lambd 0.0030806122448979595:
The training errors is 0.5869845240389064. The test error is 1.1963457560476183.
For lambd 0.003181632653061225:
The training errors is 0.6001324595729602. The test error is 1.203096071556268.
For lambd 0.0032826530612244903:
The training errors is 0.6130617468798986. The test error is 1.2098044841755542.
For lambd 0.0033836734693877555:
The training errors is 0.6257778105688381. The test error is 1.2164641444656077.
For lambd 0.003484693877551021:
The training errors is 0.6382858865043093. The test error is 1.2230693767226448.
For lambd 0.0035857142857142863:
The training errors is 0.6505910323804637. The test error is 1.2296154999172533.
For lambd 0.0036867346938775514:
The training errors is 0.6626981371520444. The test error is 1.2360986781407932.
For lambd 0.003787755102040817:
The training errors is 0.6746119295199957. The test error is 1.2425157951688128.
For lambd 0.0038887755102040822:
The training errors is 0.686336985627886. The test error is 1.248864348839178.
For lambd 0.003989795918367347:
The training errors is 0.6978777360932141. The test error is 1.2551423617905222.
For lambd 0.004090816326530612:
The training errors is 0.7092384724728252. The test error is 1.2613483057714814.
For lambd 0.004191836734693878:
The training errors is 0.7204233532423231. The test error is 1.2674810372558623.
For lambd 0.004292857142857143:
The training errors is 0.7314364093542275. The test error is 1.2735397425155075.
For lambd 0.004393877551020408:
The training errors is 0.7422815494277316. The test error is 1.279523890635272.
For lambd 0.004494897959183674:
The training errors is 0.7529625646135091. The test error is 1.2854331932216374.
For lambd 0.004595918367346939:
The training errors is 0.7634831331695467. The test error is 1.2912675697720934.
For lambd 0.004696938775510204:
The training errors is 0.7738468247780173. The test error is 1.2970271178472275.
For lambd 0.00479795918367347:
The training errors is 0.7840571046284173. The test error is 1.3027120873299654.
For lambd 0.004898979591836735:
The training errors is 0.7941173372883512. The test error is 1.3083228581730009.
For lambd 0.005:
The training errors is 0.8040307903801956. The test error is 1.3138599211313073.
```

(d)



The value of λ proposed by cross validation procedure is as following:
The lambd value with minimum cross validation errors for 5-fold is: 0.0012622448979591838.
The cross validation errors at this point is :1.4646547577624465.
The lambd value with minimum cross validation errors for 10-fold is: 0.0011612244897959184.
The cross validation errors at this point is :1.3878045664991725.

From the graph we could see that the training error is increasing when λ increases. But for the test, 5-fold and 10-fold cross validation error, we could see that the error would first decrease and then increase as λ is getting bigger. And training errors would be smaller than all the other error for all time.